Università degli Studi di Padova
Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria dell'Automazione

Tesi di laurea magistrale

iCruise: development of a smartphone app for vehicle distance estimation and tracking

Application intended to support CruiseControl devices

Candidato:
Fabio Baldo
Matricola 607371

Relatore:
Prof. Luca Schenato

Anno Accademico 2012–2013

Professore: Lei promette bene, le dicevo, e probabilmente sbaglio, comunque voglio darle un consiglio, lei ha una qualche ambizione?
Nicola: Ma... Non...
Professore: E allora vada via... Se ne vada dall'Italia. Lasci l'Italia finché è in tempo. Cosa vuol fare, il chirurgo?
Nicola: Non lo so, non... non ho ancora deciso...
Professore: Qualsiasi cosa decida, vada a studiare a Londra, a Parigi, vada in America, se ha le possibilità, ma lasci questo Paese. L'Italia è un Paese da distruggere: un posto bello e inutile, destinato a morire.
Nicola: Cioè, secondo lei tra un poco ci sarà un'apocalisse?
Professore: E magari ci fosse, almeno saremmo tutti costretti a ricostruire... Invece qui rimane tutto immobile, uguale, in mano ai dinosauri. Dia retta, vada via...
Nicola: E lei, allora, professore, perché rimane?
Professore: Come perché?! Mio caro, io sono uno dei dinosauri da distruggere.

Dialogo tratto dal film **La meglio gioventù**, del 2003 di Marco Tullio Giordana, con Luigi Lo Cascio e Alessio Boni

ABSTRACT

Negli ultimi anni i sistemi di assistenza alla guida stanno suscitando un crescente interesse nel mondo della ricerca automobilistica, questo per due motivi principali: il primo per aumentare il comfort di guida e il secondo per cercare di ridurre il numero di incidenti stradali. Essi, infatti, si pongono come obbiettivo quello di riconoscere in anticipo le situazioni di giuda critiche e aiutare a condurre in modo sicuro la vettura.
In questo contesto si inserisce il progetto *iCruise*, sviluppato dall'Università degli studi di Padova, che ha come scopo quello di sviluppare un sistema di assistenza alla giuda "Plug and Play", adattabile a qualsiasi vettura e non invasivo, cioè che non necessita di sensori esterni (radar, laser, ecc...), ma utilizza le sole informazioni ottenibili da un comune smartphone di ultima generazione posizionato sul cruscotto della vettura.
Nel seguente lavoro di tesi verrà presentata e descritta un'applicazione Android capace di fornire tutte le informazioni necessarie per poter assistere un dispositivo di Cruise Control quali velocità, riconoscimento della careggiata, individuazione degli ostacoli e stima della loro distanza, utilizzando solo i sensori presenti sullo smartphone, quali videocamera e GPS.
La sfida del progetto è quella di realizzare un'applicazione snella e fluida in grado di forniere tutte le informazioni necessarie in tempo reale e che non necessiti di un eccessivo quantitativo di memoria.

RINGRAZIAMENTI

Considero questo lavoro il capitolo conclusivo di un lungo cammino di crescita e formazione che, dal punto di vista umano, mi ha notevolmente cambiato, spero migliorandomi.
Vorrei ringraziare i miei genitori, *Ferruccio e Angela*, e mio fratello *Mattia* sia per aver creduto nella mia scelta di studi "alternativa" mantenendomi per molti anni, sia per avermi consigliato e supportato.

Un ringraziamento particolare ad Alice, un bellissimo raggio di sole in questo mare in continua tempesta!!

Ringrazio, infine, il Prof. Luca Schenato, che mi ha dato l'opportunità di realizzare questo lavoro di tesi, concedendomi la giusta libertà di scelta.

Fabio

INDICE

1 ANALISI PROSPETTICA 1
- 1.1 Modello Pin-Hole Camera 1
- 1.2 Coordinate Mondo e Coordinate Camera 3
  - 1.2.1 Bird's Eye View 5
- 1.3 Stabilizzazione della Bird's Eye View 6

2 INTRODUZIONE ALL'INDIVIDUAZIONE DEI VEICOLI 9
- 2.1 Sensori attivi versus sensori passivi 9
- 2.2 Ricerca di oggetti in immagini 10
  - 2.2.1 Hypothesis Generation (HG) 10
  - 2.2.2 Hypothesis Verification (HV) 13
  - 2.2.3 Histogram of oriented gradients 15

3 IMPLEMENTAZIONE DEL RICONOSCIMENTO DEI VEICOLI 19
- 3.1 Histogram of oriented gradients 19
- 3.2 Ricerca delle ombre 19
  - 3.2.1 Hypothesis Generation 19
  - 3.2.2 Hypothesis Verification 23

4 IMPLEMENTAZIONE 25
- 4.1 Implementazione in C++ 25
- 4.2 Implementazione in Android 25
  - 4.2.1 Calibrazione 26
  - 4.2.2 Stima della velocità 26
  - 4.2.3 Creazione dell'interfaccia 27

5 TEST E CONCLUSIONI 29
- 5.1 Algoritmo C++ 29
- 5.2 Applicazione Android 30

6 CONCLUSIONI 33
- 6.1 Sviluppi futuri 33

A APPENDICE 35
- A.1 Calibrazione della fotocamera 35
- A.2 Matrici di rotazione 36

BIBLIOGRAFIA 37

ELENCO DELLE FIGURE

Figura 1 Un esempio di immagine in prospettiva centrale 1
Figura 2 Il modello di camera pin-hole. Un punto mondo in coordinate camera viene proiettato sul piano immagine 2
Figura 3 Coordinate Immagine 3
Figura 4 Coordinate camera e coordinate mondo ISO 8855 4
Figura 5 Un esempio di *Bird's Eye View* 5
Figura 6 Esempio di *Bird's Eye View* non stabilizzata. Si noti che le linee della carreggiata non sono parallele 6
Figura 7 Un esempio di *Bird's Eye View* dove si è impostata un'altezza diversa nel vettore $\tilde{t_0}$. Come si può notare la variazione dell'altezza non modifica la *Bird's Eye View*, ma modifica solo la porzione d'immagine che viene visualizzata 7
Figura 8 Angoli di Roll Yaw e Pitch 7
Figura 9 L'individuazione di veicoli con sensori passivi è un'operazione complessa perché essi possono variare nella forma, dimensioni e colori. In più il loro "aspetto" dipende dalla loro posizione e se ci sono altri oggetti nelle vicinanze 10
Figura 10 Prima si selezionano le parti dell'immagine dove è più probabile che si trovi un veicolo e poi si verifica l'ipotesi 11
Figura 11 Geometria epipolare 13
Figura 12 Esempio di calcolo della *disparity map* 14
Figura 13 Una possibile divisione della fase del gradiente dei pixel, in questo caso $\beta = 18$ 16
Figura 14 In rosso i veicoli e in blu i non-veicoli distribuiti secondo la struttura *gradient-based* 17
Figura 15 Tutti i veicoli nell'immagine BEV presentano una caratteristica in comune: la parte scura sottostante al veicolo. 20
Figura 16 L'immagine in *Bird's Eyes View* dove si inizia la ricerca dei veicoli e estrazione della corsia 21
Figura 17 Istogramma della corsia in assenza di aree scure 21
Figura 18 Istogramma della corsia con presenza di aree scure 22
Figura 19 L'immagine della corsia *binarizzata* 19a e calcolo della distanza in pixel dell'oggetto 19b 23
Figura 20 Un esempio di area selezionata nell'immagine originale e estrazione dei bordi 24
Figura 21 Misura delle linee tratteggiate dell'autostrada 26
Figura 22 Interfaccia Android dell'applicazione 27
Figura 23 Un esempio di errori comuni nell'analisi dell'istogramma 29
Figura 24 HTC One X 30
Figura 25 Vibrazione dello smartphone 31

Figura 26 Un esempio di telecamera con comunicazione wifi, lente antiriflesso e visione notturna 34
Figura 27 Immagini utilizzate per la calibrazione 35

INTRODUZIONE

Partiamo da un dato: l'*Organizzazione delle Nazioni Unite* (ONU) stima che dal 2010 al 2020 il numero di morti per incidenti stradali, nel mondo, sarà di 1,9 milioni a causa, principalmente, dell'aumento vertiginoso del traffico nei paesi emergenti.
Governi e associazioni, pertanto, intendono incentivare e supportare lo sviluppo di sistemi che migliorano la sicurezza di guida.
L'Unione europea, per esempio, ha reso obbligatorie l'installazione sui nuovi veicoli tecnologie particolarmente efficienti come l'ABS e l'ESP.
L'*Euro NCAP* (Euro New Car Assessment Programme), l'organizzazione europea che certifica il livello di sicurezza dei nuovi veicoli, dal 2010 premia i costruttori che equipaggiano le proprie vetture con sistemi di sicurezza avanzati che offrano un beneficio tangibile agli utenti della strada. Premiando queste tecnologie, *Euro NCAP*, incentiva i produttori ad accelerare l'installazione di serie dei sistemi di sicurezza su tutti i veicoli e aiuta il consumatore a prendere la giusta decisione di acquisto.
Nello sviluppo di nuovi sistemi di sicurezza e assistenza si persegue pertanto su due strategie fondamentali: da un lato creare concreti vantaggi per il guidatore con nuovi sistemi di assistenza che rendano più sicura e confortevole la guida; dall'altro, proporre innovazioni che rendano più convenienti i sistemi esistenti in modo che possano affermarsi anche nelle auto più economiche e nei Paesi emergenti. Solo sistemi di sicurezza ampiamente diffusi possono apportare il necessario contributo all'eliminazione degli incidenti.

SISTEMI DI SICUREZZA AUTOMOBILISTICA

I dispositivi di sicurezza automobilistica vengono distinti in sistemi di sicurezza attiva e passiva e molti di essi sono, per legge, diventati obbligatori, di conseguenza forniti di serie al momento dell'acquisto di una autovettura. Alcuni dei dispositivi hanno portato ad un diverso modo di affrontare la guida di un veicolo; in un'auto munita di ABS, ad esempio, è possibile frenare a fondo anche in condizioni critiche, poiché è il sistema stesso che provvede ad evitare il bloccaggio delle ruote. Tale frenata violenta era invece assolutamente da evitare in passato, soprattutto in condizioni di asfalto bagnato o comunque di scarsa aderenza.
I dispositivi possono essere:

- Attivi, intervengono in modo da ridurre la possibilità che si verifichi un evento avverso:
  - *ABS* dispositivo che evita il bloccaggio delle ruote;
  - *ESP* congegni di controllo elettronico della stabilità dinamica, che migliora anche la funzionalità dell'ABS;
  - *TCS* riduce il pattinamento delle ruote e migliora la stabilità
- Passivi, sono protezioni che riducono le conseguenze fisiche di un danno:

- *Telaio*: i telai con le sue diverse zone di comprimibilità riesce ad assorbire parte dell'urto;
- *Cinture di sicurezza*: strumento che permette d'evitare l'eiezione del corpo fuori dall'abitacolo o l'urto con il volante e altre parti della vettura;
- *Air bag*: sistema che coadiuva il compito delle cinture di sicurezza;
- *Seggiolini*: congegni utilizzabili per la ritenuta dei bambini, dove i normali dispositivi per adulti oltre ad essere inefficaci possono essere deleteri.

SISTEMI AVANZATI DI ASSISTENZA ALLA GIUDA

Oltre ai sistemi di sicurezza "tradizionali", cioè quelli che sono presenti ormai nella maggior parte dei veicoli oggi in circolazione, esistono anche i cosiddetti *Sistemi Avanzati di Assistenza alla Giuda* (o Advanced Driver Assistance Systems). Tali sistemi sono spesso supportati da sensori "esterni" al veicolo, e sono in grado di fornire informazioni aggiuntive e/o assistenza al guidatore.
Tra i più diffusi ci sono:

- *In-vehicle navigation system*: sono navigatori GPS che, oltre a fornire le indicazioni sul percorso, forniscono anche, in tempo reale, le indicazioni sul traffico e trovano eventuali percorsi alternativi;
- *Adaptive cruise control (ACC)*: sono cruise control in grado di modificare la velocità di crociera in modo da mantenere un distanza di sicurezza dal veicolo che precede;
- *Lane departure warning system*: sono sistemi in grado di avvertire il guidatore quando il veicolo sta' iniziando a lasciare la propria carreggiata senza che sia stato attivato l'indicazione di direzione;
- *Automatic parking*: sono sistemi in grado di parcheggiare il veicolo in maniera autonoma;
- *Traffic sign recognition*: sono sistemi in grado di riconoscere la segnaletica stradale, sia orizzontale che verticale.

  Per una descrizione più dettagliata si veda *Advanced Driver Assistance Systems*.

PROGETTO ICRUISE

Nell'ambito dei *Sistemi Avanzati di Assistenza alla Giuda*, l'Università degli studi di Padova ha deciso di avviare un progetto, denominato *iCruise*, che si prefigge di progettare e realizzare un dispositivo di navigazione assistita per auto, tramite l'utilizzo di un comune smartphone di ultima generazione. In particolare il progetto richiede lo sviluppo di tre sotto-progetti:

1. lo sviluppo di un sistema software (una app) da installare sullo smartphone (Android) che, se posizionato sul cruscotto, permetta di poter stimare la velocità dell'auto e la distanza dagli ostacoli che la precedono tramite l'utilizzo della videocamera, del GPS e degli accelerometri.

2. lo sviluppo di un sistema software (una app) da installare sullo smartphone che deve controllare l'apertura della valvola motore (accelerazione) in tempo reale in base a dati forniti da 1. e alla velocità di crociera impostata dal guidatore. Tale valore deve essere poi comunicato via bluetooth ad un sistema installato sull'auto che permetta l'effettivo controllo della valvola motore. In pratica tale sistema dovrebbe mantenere la velocità dell'auto al valore preimpostato dal guidatore a meno che non si presentino ostacoli. In tal caso il sistema deve fare in modo di mantenere una distanza di sicurezza predefinita' dall'auto che la precede.

3. la progettazione e lo sviluppo di un dispositivo da installare sull'auto in grado di controllare l'accelerazione e la decelerazione del veicolo, ricevendo i valori di riferimento dallo smartphone via bluetooth.

IL CONTRIBUTO DI QUESTO LAVORO

Il seguente lavoro di tesi fà parte del sotto-progetto 1.
Il sotto-progetto 1 è stato iniziato dalla tesi Alvise, 2013, dove si è costruita l'applicazione Android da poter installare sul cellulare e si è implementata la parte di riconoscimento della carreggiata (Lane Detection).
Questo lavoro, invece, implementa la stima della velocità dell'auto attraverso l'uso del GPS, la ricerca degli oggetti (veicoli) all'interno della carreggiata e la stima della loro distanza.

Breve descrizione dei capitoli

IL PRIMO CAPITOLO offre una risposta alla seguente domanda: com'è possibile stimare la distanza di un oggetto in un immagine prospettica?

IL SECONDO CAPITOLO introduce i vari metodi presenti in letteratura per la ricerca dei veicoli in immagini;

IL TERZO CAPITOLO descrive l'algoritmo utilizzato per la ricerca veicoli;

IL QUARTO CAPITOLO descrive la parte implementativa del progetto;

IL QUINTO CAPITOLO mostra i test effettuati;

IL SESTO CAPITOLO sintetizza i risultati ottenuti e mostra possibili sviluppi futuri.

1 ANALISI PROSPETTICA

Le immagini catturate dalla videocamera dello smartphone, posta sul cruscotto della macchina, sono soggette alla cosiddetta prospettiva centrale, cioè la prospettiva avente un solo punto di fuga.
Il problema delle immagini in prospettiva è che, per esempio, un uomo in primo piano può apparire più alto di un'automobile in secondo piano o di una casa, cioè vengono eliminate tutte le informazioni riguardanti l'altezza e la profondità degli oggetti.
Per poter dunque stimare la profondità (distanza) di un oggetto in un immagine è necessario applicare delle trasformazioni geometriche in modo da poter eliminare la prospettiva.
La tecnica presa in esame è la cosiddetta *Bird's Eye View* o vista a volo d'uccello, che consente di eliminare la prospettiva nell'immagini.
Va comunque detto che in molte applicazioni per stimare la distanza di un oggetto viene utilizzata la visione stereoscopica della scena, cioè due o più telecamere riprendono l'oggetto da più punti di vista e, attraverso la geometria epipolare, si può determinare la distanza dell'oggetto. Questa tecnica non è stata presa in considerazione perché, ovviamente, il cellulare è dotato di una sola telecamera.

Figura 1: Un esempio di immagine in prospettiva centrale

1.1 MODELLO PIN-HOLE CAMERA

Introdurremo ora il processo attraverso il quale la luce incidente sugli oggetti viene impressa su un sensore digitale. Tale concetto è fondamentale nell'elaborazione delle immagini in quando fornisce la relazione che lega i punti di un'immagine con la loro posizione nel mondo, ovvero permette di determinare la zona del mondo associata a un pixel dell'immagine o, viceversa, individuare l'area dell'immagine che raccoglie una determinata regione in coordinate mondo.
Il modello proiettivo universalmente accettato, è detto *Pin-Hole Camera*, e si basa su semplici rapporti geometrici.
In figura 2 è mostrato uno schema molto semplificato di come avviene la formazione dell'immagine sul sensore.
Il punto osservato $(x_i, y_i, z_i)^T$, espresso in coordinate camera, viene proiet-

tato su una cella del sensore $(\tilde{u}_i, \tilde{v}_i)^T$. Tutti questi raggi passano per uno stesso punto: il punto focale (pin-hole).

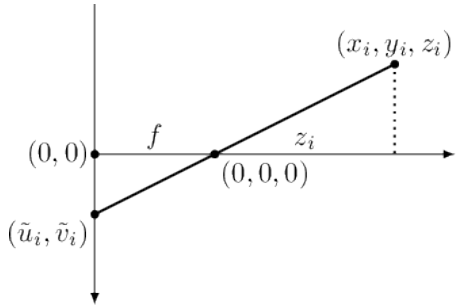


Figura 2: Il modello di camera pin-hole. Un punto mondo in coordinate camera viene proiettato sul piano immagine

Analizzando la figura 2 si vede come i rapporti tra triangoli simili, generati dai raggi ottici, descrivono l'equazione che permette di proiettare un generico punto $(x_i, y_i, z_i)^T$, espresso in coordinate camera, in coordinate sensore $(\tilde{u}_i, \tilde{v}_i)^T$:

$$\begin{bmatrix} \tilde{u}_i \\ \tilde{v}_i \end{bmatrix} = \frac{f}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{1}$$

dove f è la distanza focale (distanza tra il pin-hole e il sensore).
È da precisare che le coordinate $(x_i, y_i, z_i)^T$, espresse in coordinate camera, seguono la regola della mano sinistra (molto usata in computer graphics), contrapposta alla regola della mano destra (più usata in applicazioni robotiche) invece scelta per esprimere le coordinate mondo. L'utilizzo della coordinata z_i per esprimere la distanza è un obbligo puramente matematico a causa delle trasformazioni che verranno presentate fra poco.
Le coordinate sensore $(\tilde{u}_i, \tilde{v}_i)^T$ non sono le coordinate immagine, ma sono ancora delle coordinate "intermedie". È quindi necessario applicare una ulteriore trasformazione per ottenere le coordinate immagine:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} D_u \tilde{u}_i \\ D_v \tilde{v}_i \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} \tag{2}$$

dove le coordinate (u_0, v_0) *(principal point)* tengono conto dello scostamento dell'origine delle coordinate nell'immagine memorizzata rispetto alla proiezione del punto focale sul sensore.
D_u e D_v sono fattori di conversione tra le unità del sistema di riferimento del sensore (metri) con quelle immagine (pixel) e tengono conto dei diversi fattori di conversione coinvolti. Con l'avvento dei sensori digitali normalmente $D_u = D_v$.
In mancanza di informazioni, reperibili dai vari datasheet, su f, D_u e D_v, c'è la tendenza ad accorpare queste variabili in due nuove variabili k_u e k_v chiamate lunghezze focali efficaci (misurate in pixel), ottenibili in maniera empirica dalle immagini (si veda l'appendice Calibrazione della fotocamera).
Queste variabili, coinvolte nella conversione tra coordinate sensore e coordinate immagine, sono tra loro in relazione come:

$$\begin{aligned} k_u &= D_u f = \frac{u_0}{\tan\alpha_u} \\ k_v &= D_v f = \frac{v_0}{\tan\alpha_v} \end{aligned} \tag{3}$$

con α_u e α_v angoli approssimabili alla semi-ampiezza dell'apertura della camera (orizzontale e verticale rispettivamente).
Quando l'ottica non è distorta e il sensore ha pixel quadrati, k_u e k_v tendono ad assumere lo stesso valore.
A causa della presenza del rapporto, l'equazione 1 non è chiaramente rappresentabile in un sistema lineare. Tuttavia risulta possibile modificare tale scrittura, aggiungendo un incognita λ e un vincolo ulteriore, per poter rappresentare in forma di sistema lineare tale equazione.
Grazie alle coordinate omogenee si mostra facilmente che l'equazione 1 si può scrivere come:

$$\begin{bmatrix} \lambda u_i \\ \lambda v_i \\ \lambda \end{bmatrix} = \lambda \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{4}$$

La matrice $\mathbf{K}$, tramite 2 e 3, può essere scritta come:

$$\mathbf{K} = \begin{bmatrix} \frac{u_0}{\tan \alpha_u} & k_\gamma & u_0 \\ 0 & \frac{v_0}{\tan \alpha_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} k_u & k_\gamma & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Tale matrice, non dipendendo da fattori che non siano altri che quelli della camera stessa, è detta *matrice dei fattori intrinseci*.
Con i sensori digitali moderni e la costruzione di telecamere non manualmente, ma con macchine a controllo numerico precise, è possibile porre lo *skew factor* a zero $k_\gamma = 0$ (lo *skew factor* è un fattore che tiene conto del fatto che l'angolo tra gli assi nel sensore non sia esattamente 90 gradi).
La conoscenza di questi parametri (si veda l'appendice Calibrazione della fotocamera) determina la possibilità di trasformare un punto da coordinate camera a coordinate immagine o, viceversa, generare la retta in coordinate camera sottesa a un punto immagine.

1.2 COORDINATE MONDO E COORDINATE CAMERA

Quando si opera su problemi pratici risulta necessario passare da un sistema di riferimento solidale con la camera, dove il punto $(0,0,0)^T$ coincide con il fuoco (pin-hole) (figura 4a), a un sistema di riferimento più generico, che meglio si adatti alle esigenze dell'utilizzatore, dove la camera è posizionata in un punto generico del mondo e orientata rispetto ad esso in modo arbitrario (figura 4b).

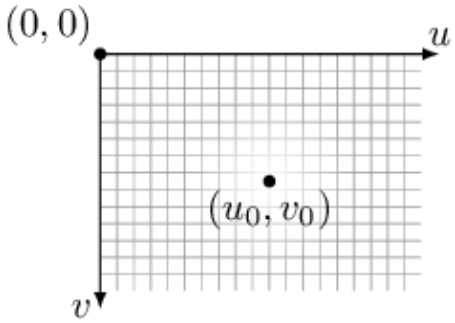


Figura 3: Coordinate Immagine

(a) Coordinate Camera **(b)** Coordinate Mondo ISO 8855

Figura 4: Coordinate camera e coordinate mondo ISO 8855

Questo discorso si applica a qualsiasi sensore generico, anche non video, definendo delle relazioni che permettono di passare dalle coordinate mondo a coordinate sensore e viceversa. Risulta altresì diffuso in letteratura usare un sistema cartesiano dove l'asse z rappresenta l'altezza del punto dal suolo piuttosto che la distanza dal pin-hole, come avviene in coordinante camera, e modificare gli altri assi di conseguenza.
Il diverso ruolo che svolge la coordinata z in coordinate camera e in coordinate mondo deriva dal fatto che, in coordinate camera la $\tilde{z}$ deve rappresentare la distanza dal pin-hole (la quantità da dividere nella proiezione prospettica), mentre, in coordinate mondo, la coordinata z rappresenta invece l'altezza.
Per arrivare all'equazione definitiva della pin-hole camera si parte dall'equazione 4 e si applicano le seguenti considerazioni:

- gli assi devono essere scambiati tra loro attraverso una trasformazione Π per ottenere il sistema di riferimento finale;
- la camera deve essere ruotata in maniera arbitraria attraverso una trasformazione $\mathbf{R}$ e, conseguentemente, non coincidere con gli assi del sistema mondo;
- il pin-hole non coincide con il punto $(0,0,0)^T$, ma giace in un generico punto $t_0 = (x_0, y_0, z_0)^T$ espresso in coordinate mondo.

Sia $(x_i, y_i, z_i)^T$ un punto in coordinate mondo e $(\tilde{x}_i, \tilde{y}_i, \tilde{z}_i)^T$ il medesimo punto in coordinate camera. La relazione che lega questi due punti si può scrivere come:

$$\begin{bmatrix} \tilde{x}_i \\ \tilde{y}_i \\ \tilde{z}_i \end{bmatrix} = \mathbf{R}\left(\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} - t_0 \right) = \mathbf{R} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} + \tilde{t}_0 \tag{6}$$

dove $\mathbf{R}$ è una matrice 3x3 che converte da coordinate mondo a coordinate camera e tiene conto delle rotazioni e della variazione del segno degli assi, mentre il vettore $\tilde{t}_0 = -\mathbf{R}t_0$ rappresenta la posizione del pin-hole t_0 rispetto all'origine del sistema mondo, rappresentato però nel sistema di coordinate camera.
Va ricordato che le matrici di rotazione sono matrici ortonormali, conservano pertanto distanze e aree, e l'inversa di una matrice di rotazione è la sua trasposta.
La matrice $\mathbf{R}$ e il vettore t_0 possono venire accorpati in forma di matrice. Grazie a questa rappresentazione, è possibile scrivere in maniera estremamente compatta la proiezione di un punto, espresso in coordinate mondo

$(x_i, y_i, z_i)^T$ in un punto di coordinate immagine $(u_i, v_i)^T$. L'equazione 7 mostra l'equazione finale del modello pin-hole (che non tiene, né deve tener conto, della distorsione):

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & | & \tilde{t_0} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{7}$$

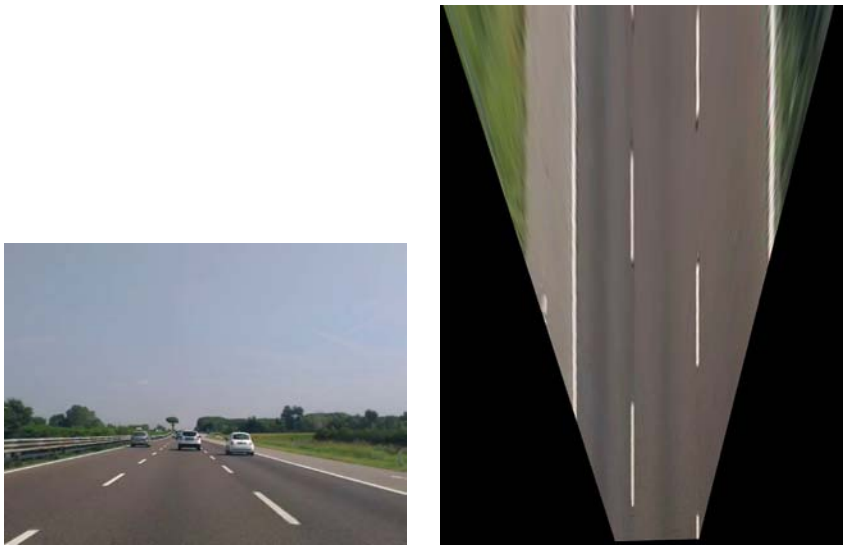
dove $\mathbf{P} = \begin{bmatrix} \mathbf{R} & | & \tilde{t_0} \end{bmatrix}$ è la *matrice proiettiva* (o camera matrix).

1.2.1 Bird's Eye View

Analizzando l'equazione 7 si nota che la matrice $\mathbf{P}$ è una matrice 3x4 e dunque non invertibile.
Ponendo, però, un vincolo aggiuntivo sui punti, e cioè $z_i = 0$, si può semplificare una colonna di $\mathbf{P}$ e, di conseguenza, la matrice diventa quadrata 3x3 e quindi invertibile. Dunque l'equazione 7 diventa:

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_i \\ y_i \\ 0 \\ 1 \end{bmatrix} = \mathbf{P}\big|_{z_i=0} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{8}$$

La matrice $\mathbf{P}\big|_{z_i=0}$ è una matrice omografica della proiezione prospettica IPM *(Inverse Perspective Mapping)* che permette di ottenere una vista dall'alto *(Bird's eye view)* della scena inquadrata. Questo perché la matrice $\mathbf{P}\big|_{z_i=0}$ permette di mappare tutti i punti con coordinata mondo $z_i = 0$, e cioè i punte che giacciono al suolo.



(a) Immagine originale **(b)** Bird's eye view

Figura 5: Un esempio di *Bird's Eye View*

La cosa importante da notare è che nella *Bird's Eye View* le linee della carreggiata risultano perfettamente parallele. Questo ci permette di calcolare la distanza di un oggetto dalla telecamera, in quanto nella *Bird's Eye View*

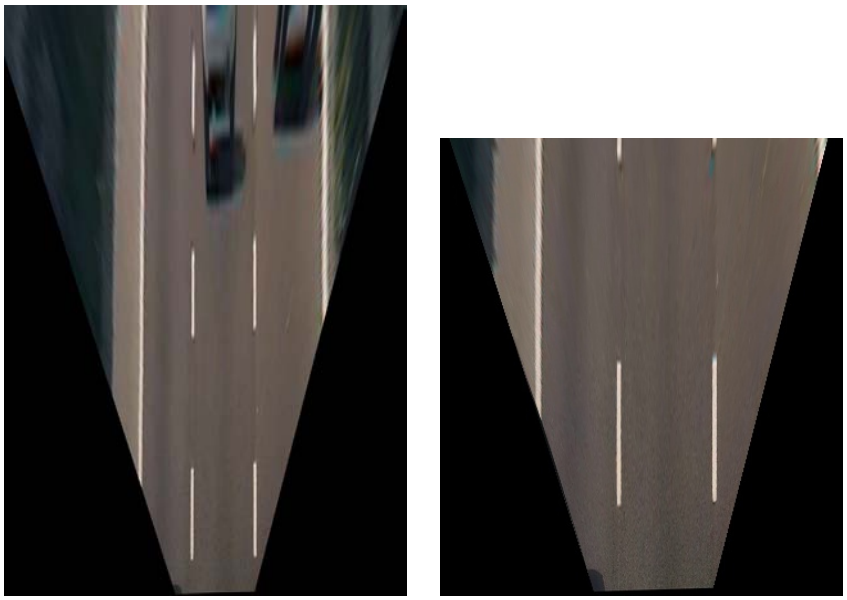
la distanza tra il punto più basso dell'immagine e l'oggetto è linearmente proporzionale alla distanza in metri dell'oggetto dalla telecamera. Cosa assolutamente non vera nell'immagine prospettica.

1.3 STABILIZZAZIONE DELLA BIRD'S EYE VIEW

Come si può notare dalle formule appena descritte per ottenere una corretta trasformazione IPM è necessario conoscere esattamente la posizione e la rotazione della telecamera rispetto alla strada.
Purtroppo però questo non può essere vero, perché la fotocamera (dello smartphone) è posta sul cruscotto della macchina e **deve** poter essere installata in posizioni diverse, in più la fotocamera non è ferma, ma subisce continue vibrazioni a causa del moto della macchina.
Tutto ciò porta ad ottenere un'errata trasformazione IPM, dove le linee della strada non sono perfettamente parallele.

Figura 6: Esempio di *Bird's Eye View* non stabilizzata. Si noti che le linee della carreggiata non sono parallele

Per risolvere queste limitazioni si è deciso di implementare un algoritmo in grado di ottenere, in modo automatico, una corretta stima della posizione e della rotazione della fotocamera rispetto al suolo.
Tale algoritmo necessita della conoscenza del punto di fuga *(o Vanishing Point VP)* dell'immagine. La stima del VP è stata implementata nel lavoro precedente (si veda il capitolo 6 di Alvise, 2013).
La conoscenza del VP è molto importante perché ci offre le informazioni riguardanti la posizione e la rotazione della fotocamera (dello smartphone) rispetto alla strada.
I parametri di traslazione (il vettore $\tilde{t_0}$), cioè la posizione della fotocamera rispetto al suolo, sono stati considerati costanti, perché essi non introducono differenze nel calcolo della trasformazione IPM, ma modificano solo la posizione della strada all'interno dell'immagine IPM (vedi figura 7).



(a) $h = 1000 \quad mm$ **(b)** $h = 500 \quad mm$

Figura 7: Un esempio di *Bird's Eye View* dove si è impostata un'altezza diversa nel vettore $\tilde{t_0}$. Come si può notare la variazione dell'altezza non modifica la *Bird's Eye View*, ma modifica solo la porzione d'immagine che viene visualizzata

Per quel che riguarda la rotazione, invece, si considerano gli angoli di *yaw* e di *pitch* della fotocamera, ed essi vengono calcolati direttamente dalle coordinante del VP. L'angolo di *roll* non viene invece considerato perché non introduce variazioni nel calcolo dalla trasformazione IPM.

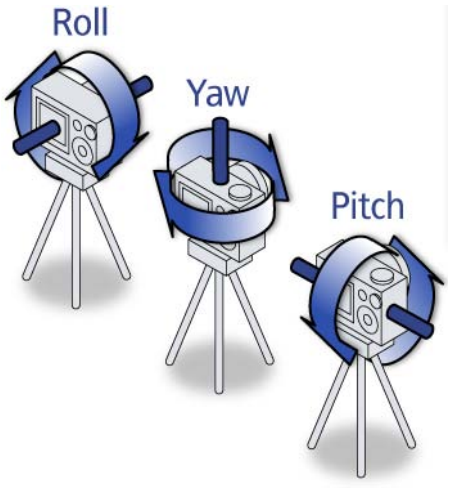


Figura 8: Angoli di Roll Yaw e Pitch

Partendo, dunque, dalla conoscenza del VP e attraverso semplici calco-

li trigonometrici, è possibile ricavare la relazione che lega il $v\bar{p} = (\bar{x}, \bar{y})^T$ all'angolo di *pitch* θ e di *yaw* γ della fotocamera:

$$\begin{aligned} \bar{y} &= \frac{N}{2}(1 - \frac{\tan\theta}{\tan\alpha_v}) \\ \bar{x} &= \frac{M}{2}(1 - \frac{\tan\gamma}{\tan\alpha_u}) \end{aligned} \tag{9}$$

dove MxN sono la larghezza e l'altezza in pixel dell'immagine.

I parametri $\tan\alpha_u$ e $\tan\alpha_v$ si possono ricavare direttamente dalla matrice dei parametri intrinseci **K**:

$$\begin{aligned} \tan\alpha_v &= -\frac{N}{2k_v} \\ \tan\alpha_v &= -\frac{M}{2k_u} \end{aligned} \tag{10}$$

Invertendo l'equazione 9 si ottengono gli angoli di *pitch* θ e di *yaw* γ della fotocamera:

$$\begin{aligned} \theta &= \arctan(\tan\alpha_v(1 - \frac{2\bar{y}}{N})) \\ \gamma &= \arctan(\tan\alpha_u(\frac{2\bar{x}}{M} - 1)) \end{aligned} \tag{11}$$

Una volta ottenuti i valori di θ e γ è possibile comporre la matrice **R** dell'equazione 8 in base alle convenzioni indicate nell'appendice Matrici di rotazione:

$$\mathbf{R} = R_\theta R_\gamma = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta)) \end{bmatrix} \begin{bmatrix} \cos(\gamma) & 0 & -\sin(\gamma) \\ 0 & 1 & 0 \\ \sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \tag{12}$$

2 INTRODUZIONE ALL'INDIVIDUAZIONE DEI VEICOLI

La ricerca autonoma di veicoli o, più in generale, ostacoli all'interno di una scena è un ambito di ricerca che sta suscitando molto interesse nell'ambiente scientifico. In questo capitolo si vuole dare una breve introduzione alle varie metodologia utilizzate per la ricerca degli ostacoli. In prima analisi verrà fatta una breve introduzione ai sensori attivi, per poi andare ad analizzare più in dettaglio i vari metodi che utilizzano i sensori passivi, ovvero videocamere. Va notato che non verranno presentati tutti i metodi presenti in letteratura, ma solo quelli che potranno essere applicati al nostro progetto, cioè i metodi utili al riconoscimento di ostacoli tramite telecamera in movimento.
Infine verrà presentata una tecnica l'*Histogram of oriented gradients*, inizialmente utilizzata per il riconoscimento di persone, ma, con opportune modifiche, utilizzata largamente anche per il riconoscimento di veicoli con ottimi risultati.

2.1 SENSORI ATTIVI VERSUS SENSORI PASSIVI

Storicamente il riconoscimento degli ostacoli all'interno di una scena viene fatto attraverso l'uso di sensori *attivi*, quali radar, laser o sensori acustici. Essi vengono definiti sensori attivi perché individuano un oggetto e ne calcolano la distanza in maniera invasiva, per trovare un oggetto devono generare un segnale e aspettare le onde riflesse dall'oggetto. La distanza dell'oggetto è proporzionale al *tempo di volo* del segnale generato, cioè il tempo tra l'istante di generazione del segnale e l'istante di arrivo del segnale riflesso dall'oggetto. Il loro vantaggio principale è che la misura della distanza è ottenuta direttamente senza utilizzo di ulteriori risorse computazionali.
Negli ultimi anni si è iniziato a utilizzare questo tipo di sensori anche sulle auto con risultati soddisfacenti. I sistemi radar, per auto, sono stati in grado di trovare ostacoli fino a 150 metri anche in condizioni di nebbia o forte pioggia, dove la visibilità è ridotta a pochi metri. I sistemi acustici, invece, sono meno costosi e più semplici da installare, tuttavia le loro prestazioni sono inferiori al radar. I sistemi laser si sono dimostrati i più precisi in assoluto, ma il loro utilizzo è limitato perché troppo costosi.
I sistemi per auto che utilizzano sensori attivi hanno mostrato ottimi risultati, tuttavia presentano due grossi svantaggi: il primo di carattere economico, il secondo è dovuto al fatto che quando un elevato numero di veicoli si muove contemporaneamente nella stessa direzione si possono creare delle interferenze tra segnali riflessi.
Il secondo approccio, nell'individuazione automatica dei veicoli, si basa sull'utilizzo di sensori *passivi*. Essi sono chiamati sensori *passivi* perché acquisiscono i dati in maniera non invasiva, i tipici sensori di questa categoria sono le normali videocamere. Il vantaggio principale di questi sensori, rispetto ai sensori attivi, è il loro basso costo e la facilità di installazione nei veicoli.

L'individuazione dei veicoli nelle immagini, però, è un'operazione complessa principalmente per l'alto numero di variabili che si devono considerare. I veicoli in circolazione, infatti, differiscono per forme, dimensioni e colori. Inoltre la forma di un veicolo nell'immagine è diversa in base all'inquadratura, una macchina inquadrata frontalmente ha una forma diversa rispetto a una inquadrata lateralmente o posteriormente. In più possono esserci altri oggetti o ostacoli nelle vicinanze che ostruiscono, anche solo parzialmente, la visuale.
Un altro fatto da considerare è che ci troviamo in un ambiente esterno, quindi ci possono essere problemi di illuminazione, di riflessione e altri fattori che complicano l'acquisizione d'immagini difficili da prevedere.

Figura 9: L'individuazione di veicoli con sensori passivi è un'operazione complessa perché essi possono variare nella forma, dimensioni e colori. In più il loro "aspetto" dipende dalla loro posizione e se ci sono altri oggetti nelle vicinanze

2.2 RICERCA DI OGGETTI IN IMMAGINI

Come si è potuto capire da questa breve introduzione, il riconoscimento di veicoli tramite sensori passivi richiede un elevato carico computazionale. Ricercare gli oggetti all'interno dell'intera immagini è spesso proibitivo per le applicazioni che devono lavorare in *real-time*. Per cercare di ridurre il carico computazionale di solito si divide il lavoro in due parti: prima si ricercano possibili porzioni dell'immagine dove si ha più probabilità di trovare un oggetto (*Hypothesis Generation HG*), poi si verifica se, nelle aree selezionate, è presente un veicolo (*Hypothesis Verification HV*).

2.2.1 Hypothesis Generation (HG)

I vari metodi HG presenti in letteratura si possono distinguere in:

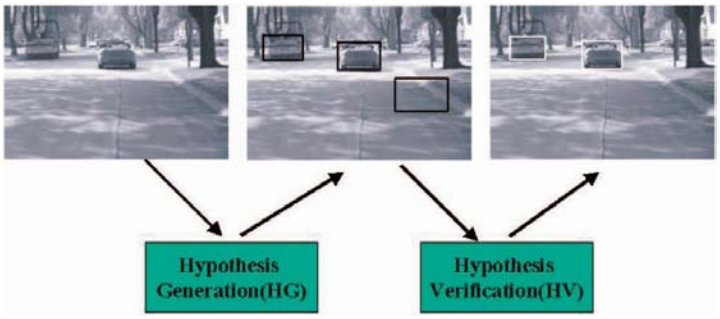


Figura 10: Prima si selezionano le parti dell'immagine dove è più probabile che si trovi un veicolo e poi si verifica l'ipotesi

- **Knowledge-based methods** utilizzano una conoscenza "a priori" per individuare possibili zone dove si possono trovare dei veicoli nell'immagine;
- **Stereo-Vision-based methods** utilizzano l'informazione da due telecamere per identificare la posizione di oggetti nell'immagini;

Knowledge-Based Methods

SIMMETRIA La simmetria è una delle principali caratteristiche degli oggetti artificiali. Le immagini che contengono veicoli inquadrati frontalmente o posteriormente presentano aree con simmetrie sia verticali che orizzontali. Questa conoscenza a priori è utilizzata come spunto per la ricerca di oggetti all'interno delle immagini. Un problema noto, di questa tecnica, è il riconoscimento come oggetti di aree omogenee presenti nell'immagine, per esempio l'asfalto o il cielo. Per eliminare queste aree è necessario filtrare l'immagine individuando prima i bordi degli oggetti e poi verificare le eventuali simmetrie.

COLORE L'informazione sul colore è utilizzata per il riconoscimento della strada in modo da poter distinguere la zona dell'immagine dove ricercare i veicoli.
Per far ciò vengono utilizzate due telecamere, una impostata con il diaframma aperto per catturare le zone più scure, l'altra con il diaframma chiuso per catturare le zone più chiare. Combinando le informazioni sul colore (i.e. Rosso, Verde e Blu) delle due telecamere si ottiene uno spazio colore $6-dimensionale$. Sapendo che questo nuovo spazio colore è distribuito secondo un modello gaussiano, è possibile costruire un classificatore in modo da poter distinguere i pixel della strada dagli altri.
Una volta identificata la strada viene perciò eliminata tutta la parte dell'immagine dove non è necessario ricercare i veicoli (tipo il cielo, i lati della strada, ecc...) limitando così notevolmente la ricerca.

OMBRA Questa tecnica viene spesso impiegata nelle ricerca di veicoli quando la telecamera è installata su un auto. Analizzando questo tipo particolare di immagini, è stato dimostrato che l'ombra del veicolo, ovvero l'area sottostante al veicolo, è distintamente più scura di ogni altra parte della strada.
Per identificare queste aree viene studiato l'istogramma dell'immagine trasformata in scala di grigi. Esso presenterà un primo picco principale in corrispondenza dei pixel della strada e un secondo picco in corrispondenza dei pixel scuri, cioè i pixel che appartengono alle ombre (del veicolo e non).

Per poter selezionare tali aree è necessario trovare un valore di soglia in grado di distinguere i pixel. Il problema è che non è possibile fissare un valore di soglia assoluto, perché l'intensità dell'ombra dipende da fattori ambientali (illuminazione, condizioni climatiche, ecc.).
Alcune applicazioni adottano il doppio valore di soglia, minimo e massimo, altre invece trovano il valore di soglia ad ogni frame, analizzando i picchi dell'istogramma.

BORDI VERTICALI/ORIZZONTALI L'immagine di un veicolo, specialmente la vista frontale o posteriore, contiene un insieme di strutture che hanno bordi orizzontali e verticali tipo parabrezza, lunotto posteriore, paraurti, ecc. Usando questa conoscenza a priori è possibile trovare delle area dove è più probabile la presenza di un veicolo.
I bordi vengono estratti utilizzando specifici operatori, come ad esempio il filtro di Sobel. Una volta estratti i bordi, si inizia la ricerca scansionando l'immagine dal basso fino ad una certa altezza predefinita, corrispondente ad una distanza massima nelle coordinate mondo. Ogni bordo orizzontale può essere un possibile candidato se e solo se è delimitato da due bordi verticali.

ANGOLI Un metodo presente in letteratura, che però non ha riscontrato molto successo, si basa sul fatto che la forma di un veicolo è delimitata da quatto angoli (in alto a sinistra, in basso a sinistra, in alto a destra e in basso a destra).
Il metodo prima trova tutti gli angoli presenti nell'immagine e poi trova le corrispondenze tra angoli in modo da formare un insieme di quadrati.

LUCI DEL VEICOLO In ultima analisi viene presentato un metodo per il riconoscimento di veicoli di notte. Molti dei metodi discussi in precedenza, infatti, sono impossibili da applicare quando la luminosità è bassa. Perché è difficile se non impossibile trovare ombre, angoli o bordi in un'immagine notturna.
Una possibile soluzione, sperimentata, è la ricerca delle luci posteriori dei veicoli. Per limitare la ricerca solo le luci che soddisfano certe condizioni di forma, dimensione e distanza minima tra di loro possono essere considerate le luci posteriori dell'auto.

Stereo-Vision-Based Methods

Il metodo presente in questo paragrafo utilizza l'informazione ottenuta da due telecamere che inquadrano la stressa scena. Anche se non potrà essere utilizzato nel lavoro di tesi (il cellulare dispone di una sola telecamera!), viene data lo stesso una breve introduzione, perché è un metodo ampiamente diffuso soprattutto in casi in cui è necessario stimare la profondità (distanza) di un oggetto.

DISPARITY MAP La visione stero è la composizione di due immagini ottenute da due telecamere che osservano la stessa scena con differenti punti di vista. Attraverso la *geometria epipolare* è possibile porre dei vincoli e delle relazioni tra il punto 3d e le rispettive proiezioni su due piani immagine delle due fotocamere. Queste relazioni ci permettono di stimare la profondità del punto nella scena.

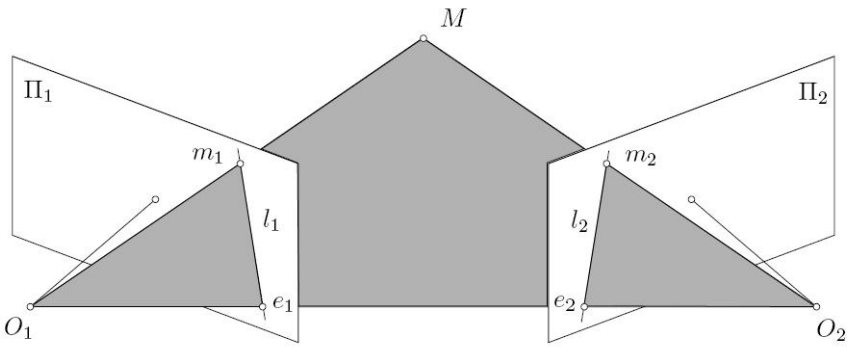


Figura 11: Geometria epipolare

Supponiamo che un punto 3d, *M*, venga osservato da due telecamere aventi rispettivamente centro ottico O_1 e O_2, otteniamo così le proiezioni m_1 e m_2 di *M* nei due piani immagine (figura 11).
Definiamo *piano epipolare* il piano che contiene M, O_1 e O_2. Notiamo che m_1 e m_2 appartengono al piano epipolare.
Consideriamo il caso dove m_2, O_1 e O_2 sono noti e vogliamo trovare il corrispondente punto m_1 nella prima immagine. Sappiamo che m_1 deve giacere sia sul piano epipolare che sul piano immagine della prima telecamera, deve quindi trovarsi sulla retta l_1, cioè la retta di intersezione dei due piani definita *retta epipolare*.
Naturalmente per simmetria si potrebbe fare il ragionamento opposto e trovare che m_2 deve giacere sulla retta l_2.
Se le due telecamere sono perfettamente calibrate, le rette epipolari risultano tutte orizzontali e si ottiene che il punto m_1 di coordinate $m_1 = [u_1 \quad v_1]^T$ e il punto $m_2 = [u_2 \quad v_2]^T$ hanno la stessa altezza $v_1 = v_2$. In questo contesto la disparità viene definita come:

$$d = u_2 - u_1 \tag{13}$$

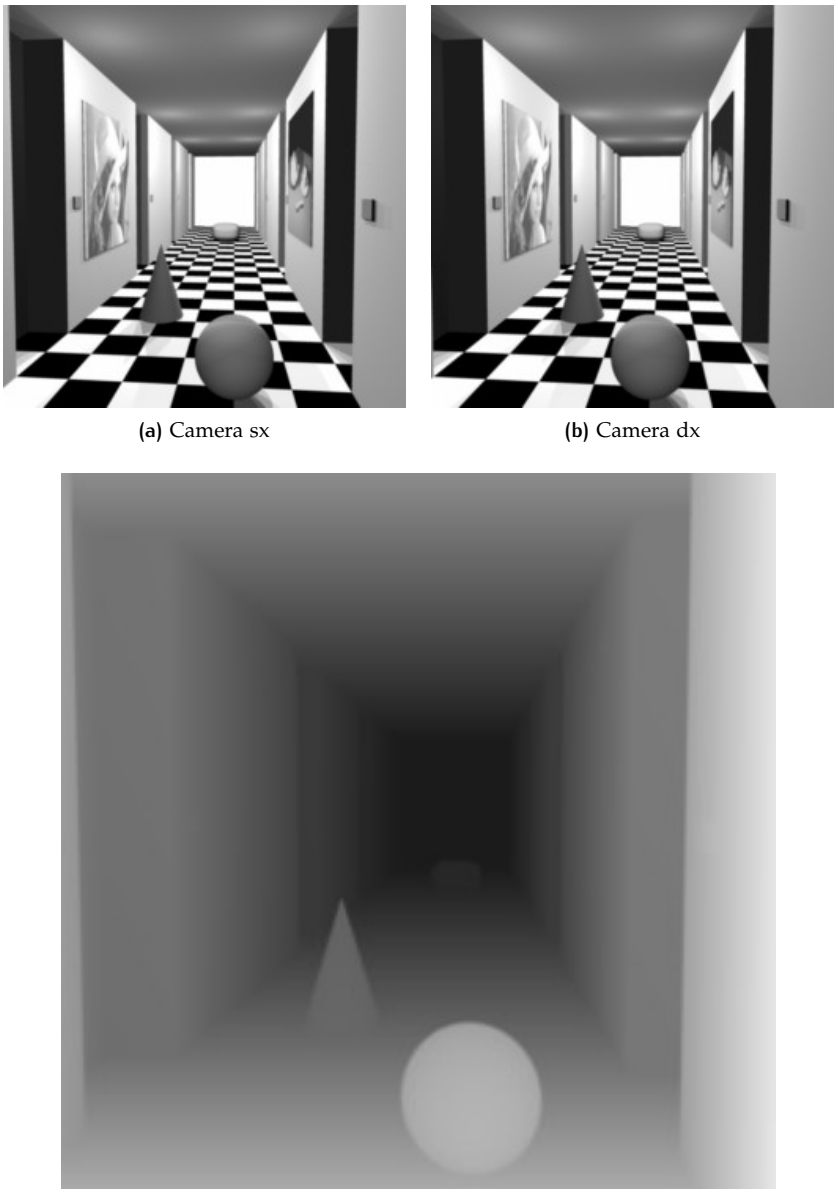
E' possibile così stimare la distanza di un punto 3d solo dalla conoscenza della sua disparità, sapendo che la distanza è inversamente proporzionale alla disparità. L'insieme di tutte le disparità si definisce mappa delle disparità (*disparity map*). In figura 12 viene mostrato un esempio di mappa delle disparità.

Questa tecnica è molto diffusa nella *computer vision* e si trovano diversi lavori in letteratura che ne mostrano vantaggi, pregi e difetti. Va osservato, infine, che nelle applicazioni il real time, per ottenere un'informazione corretta, è necessario acquisire le due immagini allo stesso istante, si deve dunque realizzare una logica di controllo in grado di sincronizzare le due telecamere.

2.2.2 Hypothesis Verification (HV)

Una volta ipotizzate le aree dov'è più probabile trovare un veicolo si deve verificare se l'ipotesi è corretta oppure no. Esistono due tipologie di verifica d'ipotesi:

- **Template-Based Methods** dove si usa un *template* predefinito del veicolo e si trova la correlazione tra l'immagine e il modello



(a) Camera sx

(b) Camera dx

(c) Disparity map

Figura 12: Esempio di calcolo della *disparity map*

- **Appearance-Based Methods** dove prima si trovano un insieme di caratteristiche comuni dei veicoli presenti in un insieme predefinito di immagini (training), per poi verificare se l'oggetto nell'immagine ha le tali caratteristiche o no.

Template-Based Methods

I *Template-based methods* usano un modello (*template*) predefinito del veicolo e ne misurano la correlazione tra l'immagine e il modello.
Purtroppo, però, non esiste uno studio che mostra l'efficienza qualitativa dei vari *templates* presentati, spesso dunque la scelta di quale usare è puramente di carattere pratico, cioè si sceglie un *tamplete* in base alle caratteristiche del proprio progetto senza conoscere l'effettiva efficienza.

Alcuni *tamplates* utilizzano delle informazioni a priori comuni su tutti i veicoli, per esempio la ricerca, all'interno dell'area selezionata, della targa o delle luci posteriori.
Altri, invece, fanno delle supposizioni di carattere più generale, per esempio il tamplate *moving edge* misura l'intensità dei bordi all'interno dell'immagine. Il tamplate si compone in due passi: prima vengono estratti i bordi dall'area selezionata attraverso opportuni operatori, poi viene calcolata la densità dei bordi. Se la densità è superiore a una certa soglia, si può supporre di aver individuato un veicolo (o comunque un oggetto artificiale!!)

Appearance-Based Methods

I metodi di verifica di ipotesi che si basano sull'*Appearance-Based Methods* si riducono ad un problema di classificazione binaria. Si tratta cioè di verificare se, nell'area selezionata, si è in presenza di un veicolo o no.
Costruire un classificatore robusto ed efficiente significa determinare un confine tra le due classi in modo da poter classificare correttamente gli oggetti. Dato l'alto numeri di variabili, costruire un classificatore non è mai un'operazione semplice. Spesso il classificatore viene costruito utilizzando un insieme di *feature* estratte da un campione di immagini contenete gli oggetti che si vogliono identificare denominato *training set*. Con questa operazione si ottiene un classificatore in grado di discriminare gli oggetti, precedentemente individuati, e dire se appartengono ad una classe o no.
Si tratta sostanzialmente di un'applicazione delle *Support Vector Machine*, ampiamente utilizzate nelle reti neurali. Per una descrizione dettagliata si veda *Support Vector Machine*.

2.2.3 Histogram of oriented gradients

L' *Histogram of Oriented Gradients (HOG)* è una tecnica di estrazione di *feature* usata in *computer vision* e in *image processing* per la ricerca e il riconoscimento di oggetti.
Tale tecnica è stata introdotta inizialmente per la ricerca di persone, o meglio pedoni, in immagini statiche da Navneet e Bill, 2005.
Successivamente si è iniziato a usare l'*HOG* anche per il riconoscimento dei veicoli, naturalmente dopo opportune modifiche. Lavori recenti hanno mostrato l'effettiva efficienza di tale tecnica rispetto alle tecniche "tradizionali". Per esempio in Jon *et al.*, 2012 viene mostrato che attraverso la tecnica *HOG* è possibile riconoscere correttamente fino al 97% dei veicoli, rispetto a un 80% di veicoli correttamente riconosciuti con la tecnica delle simmetrie. Purtroppo, però, tale tecnica richiede un elevato carico computazionale in fase di elaborazione che rende proibitivo l'uso dell'*HOG* in applicazioni che lavorano in *real time*.
Di seguito daremo una breve descrizione del'*HOG* per il riconoscimento dei veicoli.
Per prima cosa si divide l'immagine in n celle di dimensione s, e, per ogni cella, si calcola il gradiente di ogni pixel.
Il gradiente, espresso in forma vettoriale, ha fase compresa tra $[0, 180)$ gradi. Tale spazio viene suddiviso in settori uniformi (vedi figura 13).
Ogni pixel viene inserito nel corrispondente settore e si individua l'orientazione dominante (o settore dominante) o_d, cioè il settore che contiene il

maggior numero di pixel.

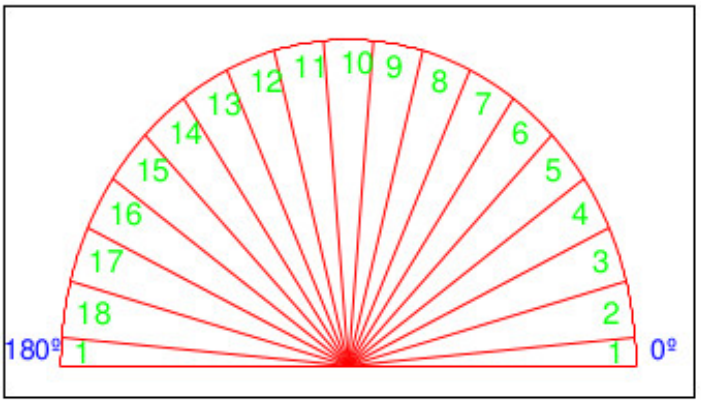


Figura 13: Una possibile divisione della fase del gradiente dei pixel, in questo caso $\beta = 18$

Si calcola poi la distanza minima tra l'orientazione dominante e l'orientazione verticale e orizzontale.

$$\begin{aligned} d_v &= d(o_d - o_v) \\ d_h &= d(o_d - o_h) \\ f_1^c &= min(d_v, d_h) \end{aligned} \tag{14}$$

Per esempio se $\beta = 18$ abbiamo che $o_v = 1$, $o_h = 10$, supponiamo di avere un'orientazione dominante del gradiente compresa tra 25 e 35 gradi, cioè $o_v = 4$, si ottiene:

$$\begin{aligned} d_v &= d(o_d - o_v) = |4 - 1| = 3 \\ d_h &= d(o_d - o_h) = |4 - 10| = 6 \\ f_1^c &= min(d_v, d_h) = min(3, 6) = 3 \end{aligned} \tag{15}$$

Successivamente si calcola il numero di celle significative, cioè le celle che hanno un numero di pixel all'interno del settore dominante maggiore di un valore di soglia t_p.
Se il numero di celle significative è nullo, vuol dire che siamo in presenza di un'area omogenea (tipo il cielo o l'asfalto) dove i pixel sono distribuiti in modo uniforme su tutti i gli insiemi. Al contrario, maggiore è il numero di celle significative, più probabile è di trovarsi in presenza di un oggetto.
La struttura finale è composta dunque da due *feature*, e cioè dal numero di celle significative f_2 e dalla distanza media tra le orientazioni f_1, calcolata come:

$$f_1 = \frac{1}{f_2} \sum_{c=1}^{n} f_1^c \tag{16}$$

con n il numero di celle.
Un esempio di come si distribuiscono i dati, usando la tecnica *HOG*, e le possibili regioni di decisione sono mostrati in figure 14.

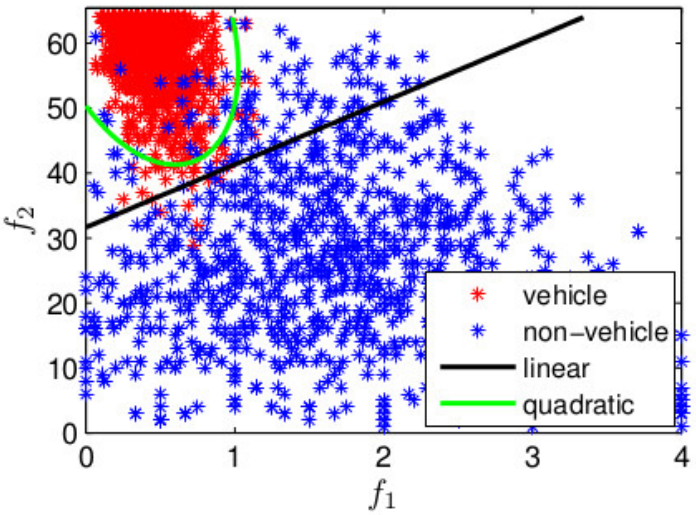


Figura 14: In rosso i veicoli e in blu i non-veicoli distribuiti secondo la struttura *gradient-based*

3 IMPLEMENTAZIONE DEL RICONOSCIMENTO DEI VEICOLI

3.1 HISTOGRAM OF ORIENTED GRADIENTS

Sull'analisi degli ottimi risultati mostrati nel lavoro di Jon *et al.*, 2012, dove con l'HOG vengono riconosciuti correttamente il 97% dei veicoli, il primo tentativo è stato quello di replicare tale tecnica in questo progetto.
La cosa nota è che si tratta di un metodo dispendioso dal punto di vista computazionale, ma, nonostante ciò, si è deciso di implementarlo comunque.
L'algoritmo è stato scritto in C++ per poter usufruire al meglio delle librerie di computer vision *OpenCV* e fatto funzionare sul PC portatile. Dopo alcune analisi si è notato che non si superavano i 5 FPS anzi, in alcuni casi, erano necessari più secondi per elaborare un singolo frame.
Per poter ovviare a questo problema e lavorare in *real time*, in letteratura è presente un lavoro (Ian e Victor Adrian, 2009) che mostra come migliorare le prestazioni dell'HOG utilizzando le librerie *CUDA*.
Tali librerie sono librerie grafiche sperimentali rilasciate da `nVidia` e permettono di effettuare calcolo parallelo sulle *GPU* della scheda video.
Attraverso l'utilizzo di queste librerie è possibile aumentare fino a 67 volte le prestazioni dell'algoritmo HOG.
Purtroppo, però, attualmente tali librerie non sono disponibili per i processori grafici nVidia installati sui moderni smartphone, ovvero i processori *Tegra 3*, ma verranno rilasciate per la prossima famiglia di processori grafici *Tegra 5*.
Sulla base di queste considerazioni si è deciso, dunque, di non utilizzare l'HOG per il riconoscimento di veicoli, ma utilizzare altre tecniche più leggere dal punto di vista computazionale.

3.2 RICERCA DELLE OMBRE

Come discusso ampiamente nel secondo capitolo, ricercare gli oggetti nell'intera immagine è un'operazione troppo dispendiosa dal punto di vista computazionale per le applicazioni che necessitano di lavorare in *real time*. Si è proceduto, pertanto, a dividere il lavoro in due parti:

- **Hypothesis Generation**: si ricercano possibili porzioni dell'immagine dove si ipotizza la presenza di un veicolo
- **Hypothesis Verification**: si verifica se, nelle aree selezionate, è presente un veicolo o no

3.2.1 Hypothesis Generation

In questa prima fase si iniziano a ricercare le aree dell'immagine dove si pensa che di poter individuare un veicolo.

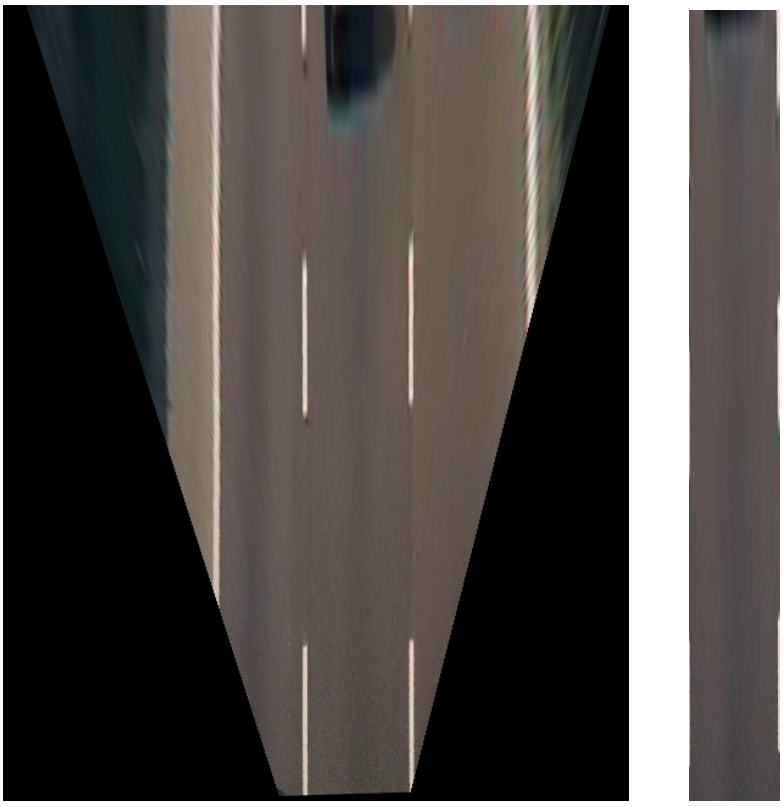
Per prima cosa è necessario eliminare tutte le parti dell'immagine che non vogliamo analizzare, come il cielo o i bordi della strada.
La cosa più semplice è quella di sfruttare l'informazione già acquisita, cioè di sfruttare la *Bird's Eyes View (BEV)*. Questo perché nell'immagine BEV vengono già escluse le parti che non ci interessano, in più non si è ritenuto necessario ricercare i veicoli che non compaiono nella BEV perché, se non sono presenti, non è possibile stimarne la distanza.
Osservando le immagini BEV si nota che tutti i veicoli mostrano una caratteristica comune. Questa caratteristica è la parte scura sottostante al veicolo composta dall'ombra, dalle ruote e dal paraurti. In figura 15 è mostrato un veicoli visto nella BEV, dove si può notare l'area scura sottostante. Lo scopo sarà quello di trovare le aree scure dell'immagine BEV per poi verificare se si tratta di un veicolo o no.

Figura 15: Tutti i veicoli nell'immagine BEV presentano una caratteristica in comune: la parte scura sottostante al veicolo.

Partendo dall'immagine 16a e sfruttando le informazioni di *Lane Detection* (implementata nel lavoro di Alvise, 2013) si inizia con il selezionare nella BEV la corsia di marcia (figura 16b), eliminando così tutta la parte della strada che non ci interessa.

Sulla parte d'immagine selezionata vengono effettuate delle operazioni di filtraggio. Per prima cosa si rimuovono le informazioni, non necessarie, sul colore trasformando l'immagine in scala di grigi. Poi si effettuano delle operazioni di erosione e dilatazione necessarie per eliminare eventuali aree scure puntiformi presenti nell'immagine.
A questo punto si rende necessario utilizzare una tecnica in grado di individuare le aree scure. Esistono vari modi più o meno complicati che permettono di effettuare tale ricerca, quello utilizzato in questo lavoro si basa sull'analisi dell'istogramma della corsia.
Prima però dobbiamo distinguere due casi:

- Assenza di aree scure;
- Presenza di aree scure.



(a) Immagine BEV **(b)** Corsia

Figura 16: L'immagine in *Bird's Eyes View* dove si inizia la ricerca dei veicoli e estrazione della corsia

Assenza di aree scure

Se nella corsia di marcia non sono presenti aree scure, l'istogramma presenterà un solo picco principale corrispondente ai pixel dell'asfalto (figura 17). In tal caso possiamo supporre che nessun oggetto è presente nella corsia, o meglio nessuna area scura è presente nella corsia e, quindi, si passa al frame successivo senza nessun altra operazione.

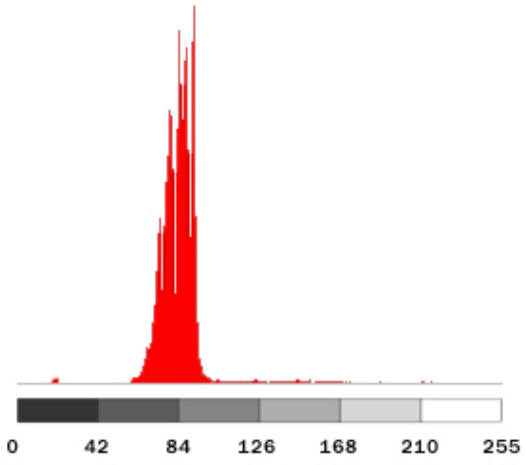


Figura 17: Istogramma della corsia in assenza di aree scure

Presenza di aree scure

Se nella corsia di marcia sono presenti aree scure, l'istogramma presenterà un picco principale corrispondente ai pixel dell'asfalto e un picco secondario corrispondente ai pixel dell'area scura (figura 18).
Per prima cosa dobbiamo individuare la posizione del secondo picco. Mentre il picco principale corrisponde al massimo assoluto dell'istogramma, il secondo picco invece non corrisponde, di norma, al secondo massimo (si veda l'immagine 18).
Per individuare il secondo picco è stata utilizzata la seguente euristica: sia h_{max_1} il valore del massimo assoluto e i la sua posizione sull'asse x, il secondo massimo sarà il punto che soddisfa la relazione:

$$h_{max_2} = max((k-i)^2 h(k)) \qquad k \in (0, 255) \tag{17}$$

In questo modo si avvantaggiano i picchi che sono più lontani del picco principale.
Individuato il secondo picco è necessario determinare una soglia S che ci permette di trasformare l'immagine da livello di grigio in immagine binaria e individuare così l'area di interesse. La soglia S non può essere fissata a priori, ma deve essere calcolata ad ogni frame, perché la posizione dei due picchi non è fissa, ma varia in base a diversi fattori ambientali.
Data la difficoltà intrinseca del problema, in letteratura esistono diversi algoritmi che consentono di calcolare la soglia di separazione ottima, partendo dalla conoscenza della posizione dei due picchi. Nel nostro caso si è sfruttato un algoritmo implementato direttamente nelle librerie *OpenCV*, ovvero il metodo di *Otsu* (o *Otsu's method*).

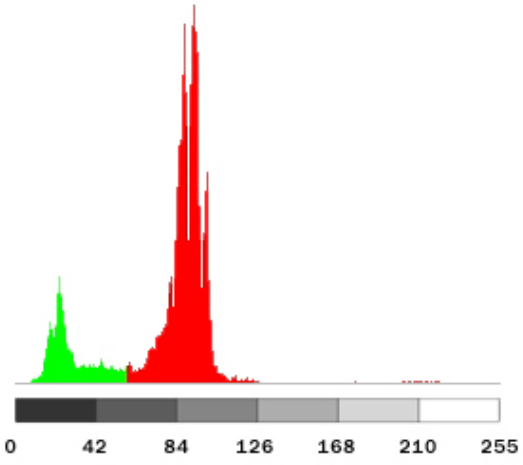
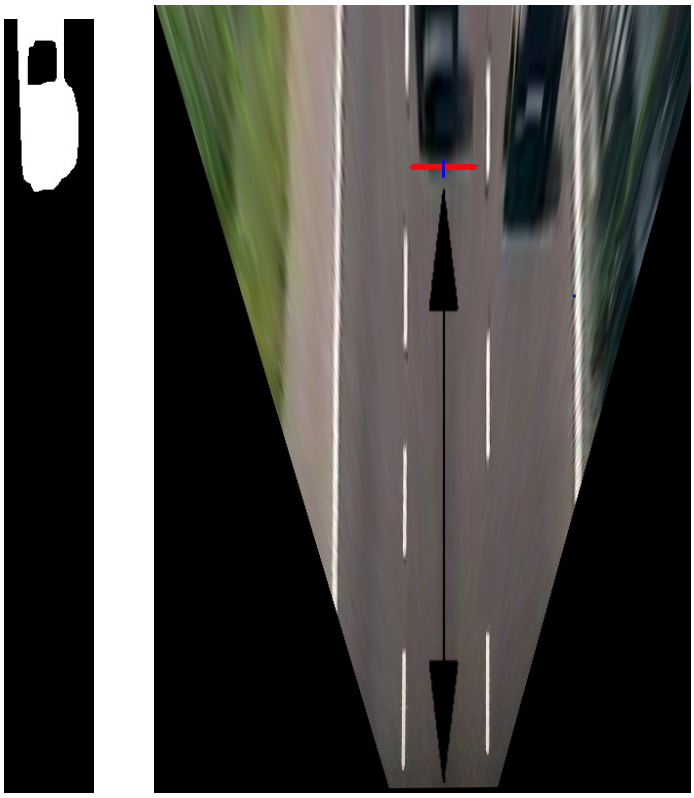


Figura 18: Istogramma della corsia con presenza di aree scure

Determinata la soglia ottima di separazione dei due insiemi di punti, è possibile *binarizzare* l'immagine della corsia e poter selezionare l'aera di interesse (figura 19a).
A questo punto si torna nell'immagine BEV e si calcola la distanza tra il punto di minimo (il punto più in basso) dell'area selezionata e il punto più in basso dell'immagine BEV (figura 19b).
Abbiamo così ottenuto la distanza in pixel dell'oggetto che corrisponde in maniera lineare alla distanza reale dell'oggetto. Verrà successivamente convertita in metri attraverso un'opportuna fase di calibrazione che sarà presen-

tata nel prossimo capitolo.

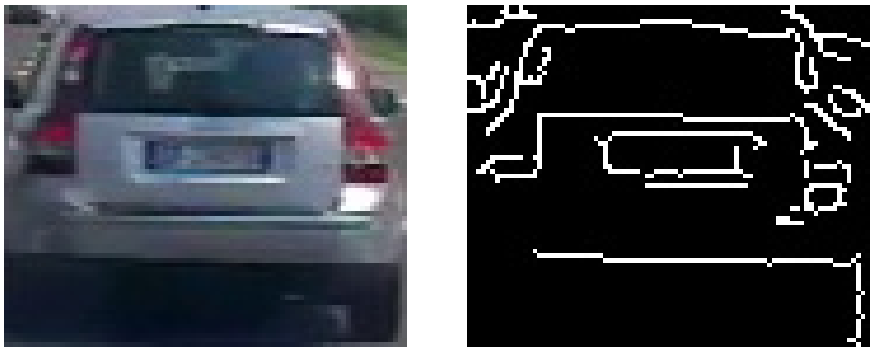


(a) Corsia **(b)** Immagine BEV

Figura 19: L'immagine della corsia *binarizzata* 19a e calcolo della distanza in pixel dell'oggetto 19b

3.2.2 Hypothesis Verification

Resta da verificare se, nell'aera selezionata, è presente un veicolo o si tratta semplicemente di un'area scura presente sulla corsia.
Con la tecnica presentata precedentemente si è determinata la parte sottostante al veicolo. Si torna a questo punto ad analizzare l'immagine originale. Da questa si estrae un'area quadrata avente come base la parte selezionata nella BEV e si ottiene un'immagine come quella mostrata in figura 20a.
Ora per determinare se si tratta di un veicolo o no, è stato usato un *Template-Based Methods* illustrato nel capitolo precedentemente, ovvero il *moving edge*. Per prima cosa si utilizza un operatore morfologico in grado di estrarre i bordi dell'immagine.
Nel nostro caso è stato usato il *Canny Edge detector* implementato nella libreria *OpenCV*. Esso ci restituisce un'immagine binaria (figura 20b) dove tutti pixel dei bordi sono posti a $p(i,j) = 1$, mentre tutti gli altri pixel sono posti a $p(i,j) = 0$.



(a) Immagine originale **(b)** Estrazione bordi

Figura 20: Un esempio di area selezionata nell'immagine originale e estrazione dei bordi

La fase successiva prevede la misura della densità dei bordi nell'immagine binaria. Essa viene calcolata dalla formula:

$$D = \frac{1}{M}\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} p(i,j) \tag{18}$$

dove M e N sono rispettivamente la larghezza e l'altezza in pixel dell'area selezionata.
La formula 18 ci restituisce un valore D che viene confrontato con un valore di soglia D_s. Dunque:

- Se $D \geqslant D_s$ si accetta l'ipotesi che nell'area selezionata è presente un veicolo;
- Se $D < D_s$ si scarta l'ipotesi.

La soglia D_s è un valore calcolato sperimentalmente. Dalle varie prove effettuate su immagini campione si è notato che le immagini di veicoli, come quella in figura 20, hanno un valore di soglia $D \in [9, 20]\%$. Si è deciso di fissare il valore di soglia $D_s = 5\%$ per avere comunque un po' di tolleranza.

4 IMPLEMENTAZIONE

L'implementazione è stata divisa in due fasi:

1. L'implementazione dell'algoritmo in C++;
2. L'implementazione dell'applicazione Android

4.1 IMPLEMENTAZIONE IN C++

Il codice dell'algoritmo è stato sviluppato in linguaggio C++ per i seguenti motivi:

- **OpenCV**: il primo, e fondamentale, motivo è stato quello di poter utilizzare appieno le potenzialità delle librerie grafiche *OpenCV*. Tali librerie sono scritte e ottimizzate per essere usate in C++, è parso ovvio, dunque, sviluppare il *core* dell'algoritmo in tale linguaggio.
  Va notato che le ultime versioni delle librerie, essendo multi-piattaforma, sono state adattate per essere utilizzate direttamente in *Android*, purtroppo però il porting su piattaforma *mobile* non è completo, nel senso che non sono state implementate tutte le funzionalità che servivano al progetto, perciò non si è potuto scrivere il codice direttamente in *Java*.
- **Debug dell'algoritmo**: il secondo motivo è dovuto ad una maggior facilità di *Debug* del codice. Visto che si sono dovuti sperimentare e testare vari algoritmi di ricerca dei veicoli, per effettuare le prove e i test necessari si è preferito lavorare sul PC con video registrati, piuttosto che sperimentare sul cellulare.
- **Porting**: l'ultimo motivo per cui si è sviluppato l'algoritmo in C++ è dovuto alla necessità di facilitare un'eventuale porting su altre piattaforme mobile, tipo *iOS* o *Windows Phone*.

L'algoritmo è composto da due moduli:

- *ipm.cpp* è il modulo che calcola la *Bird's Eyes View*;
- *distanceEstimation.cpp* è il modulo che trova gli oggetti sulla corsia e ne stima la distanza.

4.2 IMPLEMENTAZIONE IN ANDROID

Passiamo ora ad analizzare l'implementazione dell'applicazione in Android. Per poter far eseguire ad *Android* il codice scritto in C++ si è fatto uso della *Java Native Interface* o JNI. La JNI è un *framework* del linguaggio Java che consente al codice Java di richiamare (o essere richiamato) da codice cosiddetto nativo, ovvero specifico di un determinato sistema operativo o, più in generale, scritto in altri linguaggi di programmazione, in particolare C, C++ e Assembly.

La principale applicazione della JNI è quella di richiamare all'interno di programmi Java porzioni di codice che svolgono funzionalità intrinsecamente non portabili e che non posso essere implementate direttamente in Java puro. L'interfacciamento è basato sulla definizione di un insieme di classi di raccordo fra i due contesti, che presentano una interfaccia Java, ma che delegano al codice nativo l'implementazione dei loro metodi.
Va detto che la struttura dell'applicazione con le chiamate all'interfaccia nativa e tutto il resto è stata implementata nella tesi di Alvise, 2013, in questo progetto ci si è solo limitati a configurare correttamente le chiamate alla *JNI*.

4.2.1 Calibrazione

Fino a questo punto la distanza degli oggetti è sempre stata espressa in pixel, si rende necessario dunque convertire tale distanza in metri.
Un prima approccio tentato è stato quello di posizionare un veicolo ad una distanza nota e trovare la corrispettiva distanza in pixel. Purtroppo però non si sono ottenute misure attendibili.
Il secondo approccio, che ha portato ad una corretta trasformazione, è stato quello di misurare, nell'immagine in *Bird's Eyes View*, le linee tratteggiate della corsia autostradale. Tale misura, fatta su un campione di immagini, ha restituito sempre lo stesso valore: 150 pixel (figura 21).
L'articolo 40 del *codice della strada* e l'articolo 138 delle *norme di attuazione generali* prevedono che le strisce tratteggiate nelle strade dove è previsto un limite di velocità superiore a 110 km/h, siano di lunghezza pari a 4,5 m.
Grazie a questo dato e tramite una semplice proporzione siamo in grado così di convertire la misura in metri dalla distanza in pixel dell'oggetto:

$$d_{metri} = \frac{d_{pixel} \cdot 4{,}5}{150} \tag{19}$$

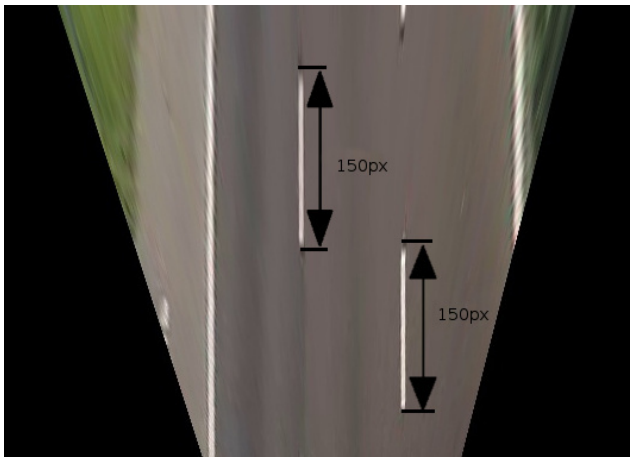


Figura 21: Misura delle linee tratteggiate dell'autostrada

4.2.2 Stima della velocità

Per la stima della velocità si sono sfruttale le *Api Location* di Android.
Le *API's location* permettono di creare applicazioni di geo-localizzazione in

modo semplice senza conoscere in dettaglio la tecnologia *GPS* di localizzazione.
Queste *API* sono ottimizzate per ridurre al minimo il consumo di energia utilizzando al meglio le capacità hardware del dispositivo.
In particolare la classe *LocationManager* mette a disposizione un metodo *getSpeed()* che restituisce la velocità al suolo espressa in metri/secondo.
Non avendo conoscenze specifiche sulle tecniche di geo-localizzazione si è deciso di utilizzare questa misura anche se non molto precisa. La realizzazione di un'applicazione di misura della velocità più precisa è un'operazione complessa che esula dagli obbiettivi di questo lavoro.

4.2.3 Creazione dell'interfaccia

L'ultima parte implementativa è stata la creazione dell'interfaccia Android. Si è optato per la realizzazione di un'interfaccia minimale che renda disponibile in maniera immediata le informazioni sulla stima della distanza e della velocità.
Come si può notare nell'immagine 22 l'interfaccia è composta in due parti:

- Nella parte sinistra viene rappresentato il frame elaborato dove attraverso una *texture* verde viene messa in evidenza la corsia di marcia, mentre con un rettangolo rosso viene inquadrato il veicolo riconosciuto.
- Nella parte destra, invece, vengono raggruppate la informazioni elaborate: misura della distanza in metri, misura della velocità espressa in km/h e il *frame rate*.

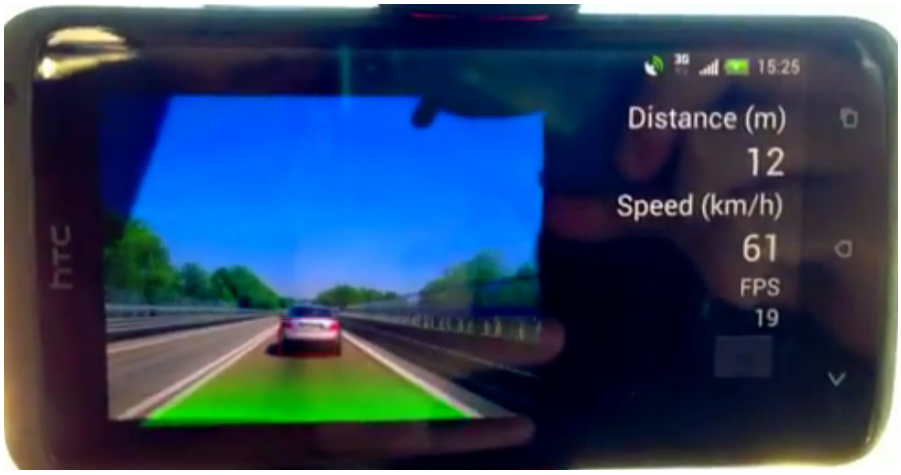


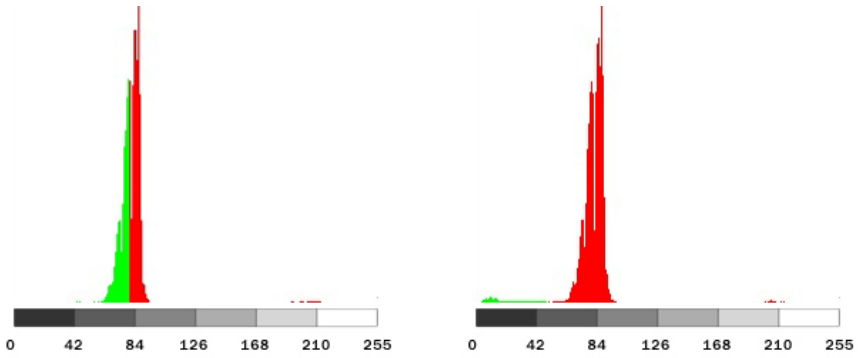
Figura 22: Interfaccia Android dell'applicazione

5 TEST E CONCLUSIONI

In questo ultimo capitolo vengono analizziate in dettaglio le prestazioni dell'algoritmo C++ e dell'applicazione Android.

5.1 ALGORITMO C++

Come detto l'algoritmo C++ è stato testato, ampiamente, su video registrati prima di essere installato su Android. Va detto per prima cosa che, per funzionare correttamente, tutti i video devono avere la stessa dimensione delle immagini con cui è stata fatta la calibrazione della fotocamera, nel nostro caso 640x480 pixel.
Video di dimensioni diverse porterebbero ad un'errata trasformazione *Bird's eyes view* dell'immagine e quindi l'impossibilità di trovare i veicolo e stimarne la distanza.
La parte dell'algoritmo che ha creato più problemi è stata l'analisi dell'istogramma. I tipici errori della valutazione dell'istogramma sono mostrati in figura 23.



(a) Errore di vicinanza **(b)** Errore di dimensione

Figura 23: Un esempio di errori comuni nell'analisi dell'istogramma

Un primo errore di analisi dell'istogramma è mostrato in figura 23a. In questo caso l'errore è dovuto al fatto che l'istogramma non ha un secondo picco significativo. La ricerca del secondo punto di massimo ha trovato un punto facente parte del primo picco significativo.
Il secondo errore è mostrato in figura 23b. Qui l'errore è dovuto al fatto che il secondo picco è troppo piccolo rispetto al primo, in questo caso infatti il secondo picco corrisponde a del rumore presente nell'immagine e non alla parte scura sottostante al veicolo.
Si è resa necessario quindi la realizzazione di una logica di controllo in grado di risolvere questi problemi noti. Essa si basa sia sull'analisi della distanza tra i due picchi e sia sull'analisi della dimensione del secondo picco rispetto al picco massimo. In questo modo si è riusciti, almeno in parte, a risolvere questi problemi. In più, visto che questi "falsi oggetti" sono

presenti in singoli frame, si è deciso di iniziare ad analizzare un oggetto solo se "compare" continuativamente per almeno 10 frame. Questa esclusione non ha portato a perdite d'informazione perché, lavorando su una media di 20 frame al secondo, 10 frame corrispondono a circa mezzo secondo.

5.2 APPLICAZIONE ANDROID

L'applicazione Android è stata installata su uno smartphone di ultima generazione messo a disposizione dall'Università: l'*HTC One X*.

Figura 24: HTC One X

L'HTC One X è animato da un processore quad-core ARM Cortex-A9 nVidia Tegra 3 da 1.5 Ghz, 1 Gb di memoria RAM, batteria da 1800 mAh, Wi-Fi IEEE 802.11 a/b/g/n, Bluetooth 4.0 con Apt-X, GPS, NFC e 32 Gb di memoria interna. Dispone inoltre di vari sensori come una Bussola digitale, Giroscopio, G-Sensor, Sensore di prossimità, Sensore di luminosità ambientale.
L'HTC One X presenta una fotocamera posteriore da 8 Mega-pixel con apertura massima di F2.0, obbiettivo grandangolare da 28 mm, singolo flash led a cinque livelli di luminosità (determinati dalla distanza dal soggetto) e con un chip immagine dedicato. La fotocamera ha un tempo di avvio di 0.7 secondi, ed una velocità di scatto di 0.2 secondi. La videocamera può registrare video fino ad una risoluzione massima di 1080p a 30 frame al secondo. È possibile anche registrare video al rallenty con risoluzione 768 x 432 pixel.
Al momento dell'utilizzo, l'HTC One X presenta Android 4.1.1 come sistema operativo e HTC Sense 4.0 come interfaccia grafica.
Si è scelto il One X principalmente perché è equipaggiato con il processore grafico nVidia *tegra 3*. Tale precessore permette di usufruire delle ottimizzazioni *tegra* delle librerie *OpenCV* che rendono le operazioni di elaborazioni fino a 20 volte più veloci.

Per i test su strada lo smartphone è stato posizionato sul cruscotto dell'auto tramite un'apposita ventosa, avendo cura di non inquadrare, con la telecamera, il cofano della macchina.
L'applicazione è stata testata per diverse ore principalmente in autostrada e in strade extraurbane principali, questo perché è in questa tipologia di strada che si ipotizza l'eventuale utilizzo del dispositivo di *Cruise Control* del progetto *iCruise*.
Non è stata condotta una approfondita analisi su quanti veicoli vengono correttamente riconosciuti perché l'applicazione, lavorando il *real time*, non registra video.
Analizziamo però pregi e difetti riscontrati nell'utilizzo su strada. Iniziamo dai difetti: il primo problema riscontrato è dovuto alle vibrazioni o a sollecitazioni esterne (buche o asfalto sconnesso) che rendono instabile lo smartphone. In questo caso si commettono degli errori perché non viene stimato correttamente il punto di fuga, viene calcolata un'errata BEV e vengono riconosciuti gli oggetti in posizioni errate. Tale problema si era, però, già riscontrato sui video registrati, come mostra la figura 25.

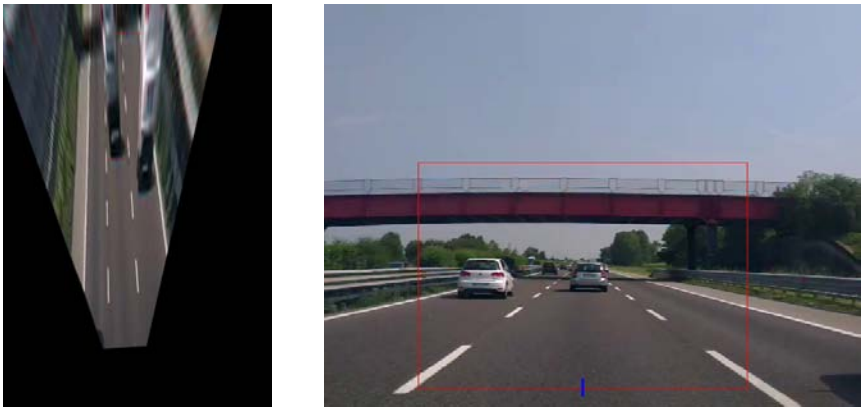


Figura 25: Vibrazione dello smartphone

L'altro problema riscontrato è dovuto all'acquisizione dell'immagine. Lo smartphone acquisisce l'immagine attraverso il parabrezza dell'auto e questo provoca problemi di riflessione quando siamo in presenza di una forte illuminazione, per esempio quando siamo controsole. Purtroppo la lente del cellulare non è dotata di trattamento antiriflesso e questo impedisce una corretta acquisizione dell'immagine in questa particolare condizione e l'impossibilità di effettuare la ricerca dei veicoli nell'immagine.
Per quel che riguarda i pregi: per prima cosa si nota la fluidità generale, l'applicazione lavora con *frame rate* compreso tra i 17 e i 22 frame al secondo, con punte massime di 26 frame al secondo quando non sono presenti veicoli nella corsia (la telecamera acquisisce le immagini a 30 frame al secondo!!) e comunque non siamo mai scesi sotto i 15 frame al secondo. In più il cellulare non risulta "eccessivamente" stressato e questo consentirà di aggiungere eventuali migliorie sia all'algoritmo che all'applicazione.

6 CONCLUSIONI

L'obbiettivo principale di questo lavoro era quello di sviluppare un'applicazione in grado di riuscire a identificare i veicoli all'interno della carreggiata e stimarne la distanza. Si può affermare di essere riusciti a raggiungere tale obbiettivo, purtroppo però l'applicazione ha mostrato dei limiti, alcuni di carattere progettuale, altri di natura intrinseca.
Gli obbiettivi che abbiamo raggiunto sono:

- **Buona misura della distanza**: tale misura soddisfa le specifiche richieste e l'errore commesso è stato valutato, sperimentalmente, inferiore ai 30 cm.
- **Corretta identificazione dei veicoli**: nelle prove effettuate i veicoli vengono riconosciuti correttamente e non si hanno mai avuto problemi di mancato riconoscimento.
- **Fluidità**: l'applicazione risulta fluida, non stressa troppo lo smartphone e il *frame rate* è sempre rimasto tra 17 e 22 fps non scendendo mai sotto i 15 fps, come ci si era prefissato.

I limiti che ci ha mostrato l'applicazione sono:

- **Distanza massima**: la distanza massima che si riesce a stimare è di 30 m. Tale distanza è proporzionale all'altezza, in pixel, massima dell'immagine BEV. Oltre tale valore non è possibile andare perché l'immagine risulterebbe troppo "strecciata" e non si riuscirebbe a riconoscere i veicoli.
- **Problemi di riflessione**: in presenza di forte illuminazione l'acquisizione delle immagini (dietro il parabrezza) risulta fortemente disturbata dai riflessi e la qualità ne viene compromessa.
- **Mancanza del riconoscimento dei pedoni**: non è stato implementato il riconoscimento dei pedoni perché non ritenuto necessario. In fase di progetto è stato ipotizzato che l'applicazione vengo usata principalmente in autostrada e/o strade extraurbane principali. Nulla vieta, però, di utilizzare tale sistema anche su strade statali/provinciali/regionali dove è probabile la presenza di pedoni.
- **Mancanza della visione notturna**: la visione notturna non è stata implementata e quindi in presenza di scarsa illuminazione l'applicazione si blocca.

6.1 SVILUPPI FUTURI

Elenchiamo ora quali potrebbero essere i possibili sviluppi futuri:

- **Risoluzione di problemi di riflessione**: sicuramente cercare di risolvere i problemi di riflessione!!
  A questo proposito si potrebbe verificare se basta l'applicazione di una

Figura 26: Un esempio di telecamera con comunicazione wifi, lente antiriflesso e visione notturna

pellicola antiriflesso sulla lente della fotocamera, oppure se è necessario dotare il progetto *iCruise* di una telecamera ausiliaria, tipo quella mostrata in figura 26. Tale soluzione porterebbe a molti vantaggi (la calibrazione verrebbe fatta una volta sola, diminuzione del carico computazionale richiesto allo smartphone, stabilizzazione dell'immagine, possibilità di acquisire immagini in notturna), ma si scontrerebbe contro la "filosofia" del progetto *iCruise*.

- **Visione notturna**: l'implementazione della visione notturna iniziando prima con la scelta di un algoritmo ad-hoc, tipo gli algoritmi che ricercano le luci posteriori o i riflessi provocati dalla targa posteriore dell'auto, per poi risolvere i problemi di messa a fuoco dell'immagine in condizioni di scarsa luminosità.
- **Porting**: infine effettuare il *porting* dell'applicazione su altre piattaforme mobile, tipo *iOS* o *Windows Phone*.

A APPENDICE

A.1 CALIBRAZIONE DELLA FOTOCAMERA

La fase di calibrazione della fotocamera permette di ricavare opportuni parametri che permettono al modello pin-hole di poter essere utilizzato per proiettare punti da coordinate mondo a coordinate camera.
Le tecniche di calibrazione si possono dividere in due categorie a seconda di quale modello della camera pin-hole si vuole ricavare:

- **implicita** dove vengono estratti gli elementi della matrice proiettiva **P** o la matrice omografica **H** in modo da poter proiettare punti da un sistema di coordinate a un altro;

- **esplicita** dove vengono estratti i parametri fisici del sistema coinvolti nella proiezione prospettica, solitamente viene calcolata la matrice **K**.

Fortunatamente la libreria *OpenCV* mette a disposizione delle funzioni che effettuano automaticamente la calibrazione implicita e/o esplicita della fotocamera, semplicemente analizzando delle foto (fatte dalla fotocamera che si intende calibrare) contenenti un "pattern" prestabilito, come, per esempio, una comune scacchiera.
Nella documentazioni *OpenCV 2.4.6.0 on-line documentation* si trova un *tutorial* che spiega dettagliatamene come eseguire la calibrazione. Alcune immagini che abbiamo utilizzato sono mostrate in figura 27:

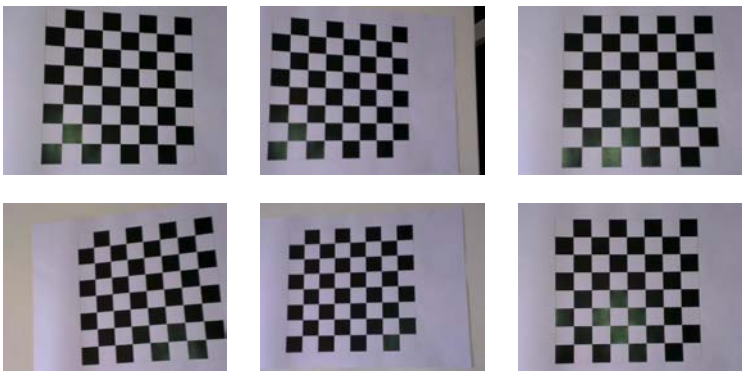


Figura 27: Immagini utilizzate per la calibrazione

Le foto sono tutte in formato 640x480, che è il formato che viene utilizzato nel progetto, e la matrice **K** calcolata dalla calibrazione è la seguente:

$$\mathbf{K} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1115.68 & 0 & 319.50 \\ 0 & 1115.68 & 239.50 \\ 0 & 0 & 1 \end{bmatrix} \tag{20}$$

A.2 MATRICI DI ROTAZIONE

Un modo per definire la matrice di rotazione in 3 dimensioni consiste nel comporre tra loro le rotazioni rispetto ai 3 assi principali del sistema di riferimento.
Definiamo θ l'angolo di *pitch* (o beccheggio), γ l'angolo di *yaw* (o imbardata) e ρ l'angolo di *roll* (o rollio), gli angoli di orientazione del sensore rispetto al sistema di riferimento. Tali angoli e tale nomenclatura sono definiti come *Tait-Bryan Angles*, o *Cardan Angles* (da Girolamo Cardano).
Di seguito saranno mostrate le matrici che convertono un vettore da coordinate sensore a coordinate mondo. Le medesime matrici possono ruotare un vettore in senso antiorario (*counterclockwise rotation of axes*) rispetto ai vari assi del sistema di riferimento:

$$\begin{aligned} R_\rho &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\rho) & -\sin(\rho) \\ 0 & \sin(\rho) & \cos(\rho) \end{bmatrix} \quad R_\theta = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \\ R_\gamma &= \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{21}$$

o in senso orario (*clockwise rotation of axes*):

$$\begin{aligned} R_\theta &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad R_\gamma = \begin{bmatrix} \cos(\gamma) & 0 & -\sin(\gamma) \\ 0 & 1 & 0 \\ \sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \\ R_\rho &= \begin{bmatrix} \cos(\rho) & \sin(\rho) & 0 \\ -\sin(\rho) & \cos(\rho) & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{22}$$

Normalmente si predilige l'utilizzo del sistema antiorario (21) in robotica o in aeronautica, invece quello orario (22) per i veicoli terrestri e per le navi. Per una spiegazione più dettagliata si veda Richard e Andrew, 2004.

BIBLIOGRAFIA

ADAS-wikipedia

Advanced Driver Assistance Systems, http://en.wikipedia.org/wiki/Advanced_driver_assistance_systems. (Citato a p. xiv.)

Alvise, Rigo

2013 *iCruise: development of a smartphone app for lane detection*, Università degli studi di Padova. (Citato alle p. xv, 6, 20, 26.)

Developers, Android

Api Location, https://developer.android.com/google/play-services/location.html. (Citato a p. 26.)

Ian, Reid e Prisacariu Victor Adrian

2009 "fastHOG - a real-time GPU implementation of HOG", *University of Oxford, Department of Engineering Science*. (Citato a p. 19.)

Jon, Arróspide, Salgado Luis e Marinas Javier

2012 "HOG-like Gradient-based Descriptor for Visual Vehicle Detection", *Intelligent Vehicles Symposium*. (Citato alle p. 15, 19.)

Marcos, Nieto, Salgado Luis, Jaureguizar Fernando e Cabrera Julian

2007 "Stabilization of Inverse Perspective Mapping Images based on Robust Vanishing Point Estimation", *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*.

Navneet, Dalal e Triggs Bill

2005 "Histograms of Oriented Gradients for Human Detection", *INRIA*. (Citato a p. 15.)

nVidia

CUDA, http://www.nvidia.com/object/cuda_home_new.html. (Citato a p. 19.)

OpenCV

OpenCV, http://opencv.org/. (Citato a p. 35.)

OpenCV 2.4.6.0 on-line documentation, http://docs.opencv.org/index.html. (Citato a p. 35.)

OTSU-Wikipedia

Otsu's method, http://en.wikipedia.org/wiki/Otsu's_method. (Citato a p. 22.)

Paolo, Medici

2013 *Elementi di analisi per Visione Artificiale*, http://www.ce.unipr.it/people/medici/geometry/index.html.

Richard, Hartley e Zisserman Andrew

2004 *Multiple View Geometry in Computer Vision*, Cambridge University Press. (Citato a p. 36.)

SVM-wikipedia

Support Vector Machine, http://en.wikipedia.org/wiki/Support_vector_machine. (Citato a p. 15.)

Zehang, Sun, Bebis George e Miller Ronald

2006 "On-Road Vehicle Detection: A Review", *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 28, NO. 5*.